Pergamon

Computers Math. Applic. Vol. 28, No. 6, pp. 33–44, 1994

0898-1221/94 $7.00 + 0.00

0898-1221(94)00150-2

A Feasible Descent Cone Method For Linearly Constrained Minimization Problems

E. DE KLERK AND J. A. SNYMAN
Structural Optimization Research Group
Department of Mechanical and Aeronautical Engineering
University of Pretoria, Pretoria 0001, South Africa

(Received January; accepted February 1994)

Abstract—An improvement over an earlier feasible directions minimization algorithm is presented. In a certain sense the new feasible descent cone algorithm is shown to be a generalization of Rosen's gradient projection method. The algorithm is evaluated for linear programming test problems, and promising results are obtained for random problems and problems involving weight minimization of plane trusses.

1. INTRODUCTION

Introduced in 1960, the gradient projection method of Rosen [1,2] remains an important theoretical cornerstone for constrained nonlinear programming. Although no positive computer results were reported for the original algorithm, the underlying ideas were implemented in more efficient algorithms [3]. More recently, interest in the gradient projection method was revived when convergence proofs for linearly constrained problems were published [4,5].

The aim of this paper is to present a generalization of Rosen's gradient projection method for linearly constrained problems. The new algorithm is called a Feasible Descent Cone (FDC) method, and is an extension of the linear programming (LP) algorithm presented by Snyman in [6]. The extension is done by following strategies analogous to those employed in the method of Rosen. The resulting algorithm gives a scheme for generating feasible search directions, and it is shown that the search directions of the gradient projection method constitute a special case of this new scheme. The FDC algorithm has the advantage that the feasible search directions may be chosen to traverse either the interior or the boundary of the feasible polytope. This is in principle an advantage over strictly boundary following methods such as Rosen's method. Promising computational results are obtained for randomly generated LP problems and for some LP problems involving the weight minimization of plane trusses.

Although the FDC method is presented here as a LP algorithm, it may readily be extended to nonlinear programs. Snyman and Stander [7] applied a simplified version of the FDC direction finding map successfully to nonlinear programming problems in structural optimization. Moreover, feasible direction methods have in the past been incorporated into hybrid methods, e.g., into the simplex [8] framework, like the method of Zangwill [9]. An evaluation of some of these algorithms is given in [10]. It is therefore hoped that the ideas presented here will find a broader field of application than LP, where recent research have been dominated by polynomial-time algorithms since the publication of Karmarkar's algorithm [11] in 1984. For a review of recent developments in the latter field the reader is referred to Roos [12].

2. THE LP PROBLEM

We formulate the FDC algorithm for the linear programming case. Once the algorithm for LP is established, the theoretical extension to nonlinear objective functions is done by simply following the analogy with Rosen's method. The LP problem is given as

$$\text{minimize } \mathbf{c}^{\top}\mathbf{x}, \tag{1}$$

subject to

$$(\mathbf{a}^i)^{\top}\mathbf{x} = b_i, \quad i = 1, 2, \ldots, n_e \in I \tag{2}$$

and

$$(\mathbf{a}^j)^{\top}\mathbf{x} \leq b_j, \quad i = n_e + 1, \ldots, n_e + n_i \in J, \tag{3}$$

where $\mathbf{c}$, $\mathbf{x}$, $\mathbf{a}^i \in \mathbb{R}^n$, $b_i \in \mathbb{R}$, I and J are sets of indices, the superscript $\top$ denotes the transpose and n_e and n_i represent the number of equality and inequality constraints, respectively. The constraints (2) and (3) define a feasible polytope. We assume every feasible point to be a regular point.

3. METHODS OF FEASIBLE DIRECTIONS

Like Rosen's method, the FDC algorithm falls in the class of feasible direction methods. Algorithms in this class proceed from a feasible point $\mathbf{x}^k$ to a new feasible point $\mathbf{x}^{k+1}$ by taking a nonzero step in a search direction $\mathbf{s}$, i.e.,

$$\mathbf{x}^{k+1} = \mathbf{x}^k + h\mathbf{s},$$

where $h > 0$ is the step length. Such a step represents one iteration of the algorithm. If $\mathbf{c}^{\top}\mathbf{x}^{k+1} < \mathbf{c}^{\top}\mathbf{x}^k$, then $\mathbf{s}$ is called a feasible descent direction. In stating the conditions for feasible search directions, we will use notation similar to that normally used in describing Rosen's method in order to draw the parallel between Rosen's method and the new method throughout.

DEFINITION 1. *The working set $W(\mathbf{x}^k)$ at the current feasible point $\mathbf{x}^k$ is defined as the index set of the m constraints ($m \leq n - 1$) that are used in calculating the search direction from $\mathbf{x}^k$. This set will always contain the set I of equality constraints.*

In Rosen's method, $W(\mathbf{x}^k)$ is the set of constraints which are active at $\mathbf{x}^k$. We assume, without loss of generality, that the $m - n_e$ inequality constraints in $W(\mathbf{x}^k)$ correspond to the constraints numbered $n_e + 1, \ldots, m$ in J.

DEFINITION 2. *$\mathbf{A}^k$ is the $m \times n$ matrix with rows given by the gradients of the m constraints in $W(\mathbf{x}^k)$. In other words, the row vectors are given by $(\mathbf{a}^i)^{\top}$ with $i \in W(\mathbf{x}^k)$.*

The conditions for feasible search directions can now be stated as follows [13]: Given a current feasible point $\mathbf{x}^k$ with working set $W(\mathbf{x}^k)$ and associated matrix $\mathbf{A}^k$, a search direction $\mathbf{s}$ is feasible provided

$$(\mathbf{a}^i)^{\top}\mathbf{s} = 0, \quad i \in W(\mathbf{x}^k) \cap I \tag{4}$$

and

$$(\mathbf{a}^i)^{\top}\mathbf{s} \leq 0, \quad i \in W(\mathbf{x}^k) \cap J \text{ and } (\mathbf{a}^i)^{\top}\mathbf{x}^k = b_i. \tag{5}$$

The conditions (4) and (5) ensure that a nonzero step may be taken from $\mathbf{x}^k$ in the direction of $\mathbf{s}$ without violating the active constraints in $W(\mathbf{x}^k)$. The method of Rosen uses the feasible descent direction given by the orthogonal projection of the steepest descent direction onto the null space of $\mathbf{A}^k$, denoted by $N(\mathbf{A}^k)$. In this case $\mathbf{s}$ is given by [1]

$$\mathbf{s} = -[\mathbf{I} - (\mathbf{A}^k)^{\top}[\mathbf{A}^k(\mathbf{A}^k)^{\top}]^{-1}\mathbf{A}^k]\,\mathbf{c}\,. \tag{6}$$

This search direction satisfies $\mathbf{A}^k\mathbf{s} = \mathbf{0}$ and therefore also satisfies (4) and (5).

4. SEARCH DIRECTIONS OF THE NEW METHOD

As before, let $W(\mathbf{x}^k)$ be the working set at the current feasible point $\mathbf{x}^k$, with $\mathbf{A}^k$ the associated matrix. Using the notation of Snyman [6], we define $\mathbf{p}^1 = -\mathbf{c}/\|\mathbf{c}\|$ and $\mathbf{p}^i = -\mathbf{a}^{i-1}/\|\mathbf{a}^{i-1}\|$, $i = 2, \ldots, m+1$, where $\mathbf{a}^i$ $(i = 1, \ldots, m)$ are the m rows of $\mathbf{A}^k$. The new method requires that the search direction $\mathbf{s}$ from $\mathbf{x}^k$ satisfies

$$(\mathbf{p}^{i+1})^\top \mathbf{s} = 0, \quad i = 1, 2, \ldots, n_e \tag{7}$$

and

$$(\mathbf{p}^{i+1})^\top \mathbf{s} = e_{i+1} \geq 0, \quad i = 0, n_e + 1, n_e + 2, \ldots, m, \tag{8}$$

where

$$\mathbf{s} = \sum_{i=1}^{m+1} \gamma_i \mathbf{p}^i. \tag{9}$$

Note that conditions (7) and (8) ensure satisfaction of the feasible direction conditions (4) and (5) if $W(\mathbf{x}^k)$ includes the set of active constraints at $\mathbf{x}^k$. The choice $e_1 > 0$ ensures descent. Utilizing expression (9), the equations (7) and (8) may be combined as a single matrix equation

$$\begin{bmatrix} (\mathbf{p}^1)^\top \mathbf{p}^1 & (\mathbf{p}^1)^\top \mathbf{p}^2 & \cdots & (\mathbf{p}^1)^\top \mathbf{p}^{m+1} \\ (\mathbf{p}^2)^\top \mathbf{p}^1 & (\mathbf{p}^2)^\top \mathbf{p}^2 & \cdots & (\mathbf{p}^2)^\top \mathbf{p}^{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{p}^{m+1})^\top \mathbf{p}^1 & (\mathbf{p}^{m+1})^\top \mathbf{p}^2 & \cdots & (\mathbf{p}^{m+1})^\top \mathbf{p}^{m+1} \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ \vdots \\ \gamma_{m+1} \end{bmatrix} = \begin{bmatrix} e_1 \\ 0 \\ \vdots \\ 0 \\ e_{n_e+2} \\ \vdots \\ e_{m+1} \end{bmatrix},$$

or

$$\mathbf{P}^k \boldsymbol{\gamma} = \mathbf{e}, \tag{10}$$

with $e_1 > 0$ and $e_{j+1} > 0$, $\forall j \in J$. Without loss of generality, we choose e_1 constant throughout as $e_1 = 1$. Snyman [6] used $e_1 = 0$, which results in a projection of $\mathbf{s}$ onto the plane of constant function value at the current feasible point. This is an unnecessary restriction and does not allow a descent step.

The zeros in the vector $\mathbf{e}$ in (10) correspond to the equality constraints contained in $W(\mathbf{x}^k)$. Note that $\mathbf{P}^k$ is associated with $W(\mathbf{x}^k)$. Given $W(\mathbf{x}^k)$, the new search direction $\mathbf{s}$ can in principle be obtained by solving for $\boldsymbol{\gamma}$ in (10) if $\mathbf{P}^k$ is nonsingular, and by subsequently using (9) to give $\mathbf{s}$. Alternative, equivalent formulae for $\mathbf{s}$ will be derived later for the purposes of numerical computation.

5. ASSEMBLY OF THE WORKING SET

The idea behind the new method, and with Rosen's method, is to systematically assemble working sets of constraints until an optimal set of active constraints has been identified and the Kuhn-Tucker conditions are satisfied. The new method starts at a feasible point $\mathbf{x}^0$ with the initial working set $W(\mathbf{x}^0) = I$. The method subsequently generates a sequence of points $\mathbf{x}^k$ $(k = 1, 2, \ldots)$ with associated working sets $W(\mathbf{x}^k)$. Initially, a feasible search direction $\mathbf{s}$ is generated at $\mathbf{x}^0$, by using (10) and (9), and a step is taken in the direction of $\mathbf{s}$ until the nearest constraint is met at $\mathbf{x}^1$. This constraint is now added to $W(\mathbf{x}^0)$ to give $W(\mathbf{x}^1)$. This process is repeated to generate further points in the sequence. A new search direction from a general $\mathbf{x}^k$ can be computed each time using (10) and (9) provided the matrix $\mathbf{P}^k$ remains nonsingular and $W(\mathbf{x}^k)$ contains fewer than n constraints. Note that all the inequality constraints contained in $W(\mathbf{x}^k)$ are not necessarily active at $\mathbf{x}^k$. If $W(\mathbf{x}^k)$ contains n constraints with $\mathbf{A}^k$ of full rank,

the vertex of intersection of these constraints is found. The algorithm then proceeds from this vertex if it is feasible and improving but not optimal. Strategies to deal with singularity of $\mathbf{P}^k$ will be discussed later. It is insightful, however, to first explore the relation between the search directions of the new method and those of Rosen's method.

6. INTERPRETATION OF THE SEARCH DIRECTIONS

Assume that $W(\mathbf{x}^k)$ is given and that the nonzero components of $\mathbf{e}$ in (10) are chosen from the interval $[0, 1]$. The set of search directions given by (9) and (10) for all possible choices of $\mathbf{e}$ from this interval now define a cone of feasible descent directions, provided that the set of active constraints is contained in $W(\mathbf{x}^k)$. In general, only the last encountered constraint will be active. This guarantees feasibility of the next search direction by (8), since $W(\mathbf{x}^k)$ will then contain the only active inequality constraint. If this assumption does not hold true, however, the working set can be replaced by the set of active constraints at $\mathbf{x}^k$.

If the nonzero components are chosen as equal to 1, the search direction is the centreline of the feasible cone. Conversely, if all the components of $\mathbf{e}$ except $e_1 = 1$ are chosen as zero, the resulting search direction $\mathbf{s}$ is the projected steepest descent direction (6) of Rosen's method, as is shown in the following theorem.

THEOREM 1. *If the vector* $\mathbf{e} = [1\ 0 \cdots 0]^\top$ *is used in (10), then the search direction given by (9) is the orthogonal projection of* $-\mathbf{c}$ *onto the null space of* $\mathbf{A}^k$ *(see (6)), under the following two assumptions:*

(i) The matrix $\mathbf{A}^k$ *is of full rank;*

(ii) $\mathbf{c} \notin \mathfrak{R}(\mathbf{A}^k) \equiv$ *row space of* $\mathbf{A}^k$.

PROOF. For simplicity and without loss of generality, the matrix $\mathbf{A}^k$ will be replaced in the following steps by

$$\mathbf{A} = \left[\frac{\mathbf{a}^1}{\|\mathbf{a}^1\|} \cdots \frac{\mathbf{a}^m}{\|\mathbf{a}^m\|} \right]^\top = - \left[\mathbf{p}^2 \cdots \mathbf{p}^{m+1} \right]^\top,$$

where the $\mathbf{a}^i$'s are the gradients of the m working set constraints. The matrix $\mathbf{A}$ is row equivalent to $\mathbf{A}^k = [\mathbf{a}^1 \cdots \mathbf{a}^m]^\top$ and therefore has the same row space and null space. We also assume (again without loss of generality) that $\|\mathbf{c}\| = 1$ so that $-\mathbf{c} = \mathbf{p}^1 = -\mathbf{c}/\|\mathbf{c}\|$. It follows from (9) that the direction finding map is now given by

$$\mathbf{s} = -[\mathbf{c}, \mathbf{A}^\top]\boldsymbol{\gamma}, \tag{11}$$

where

$$\mathbf{P}^k \boldsymbol{\gamma} = \mathbf{e}$$

as before. The matrix $\mathbf{P}^k$ may be rewritten as

$$\mathbf{P}^k = \begin{bmatrix} (\mathbf{p}^1)^\top \mathbf{p}^1 & (\mathbf{p}^1)^\top \mathbf{p}^2 & \cdots & (\mathbf{p}^1)^\top \mathbf{p}^{m+1} \\ (\mathbf{p}^2)^\top \mathbf{p}^1 & (\mathbf{p}^2)^\top \mathbf{p}^2 & \cdots & (\mathbf{p}^2)^\top \mathbf{p}^{m+1} \\ \vdots & \vdots & \ddots & \vdots \\ (\mathbf{p}^{m+1})^\top \mathbf{p}^1 & (\mathbf{p}^{m+1})^\top \mathbf{p}^2 & \cdots & (\mathbf{p}^{m+1})^\top \mathbf{p}^{m+1} \end{bmatrix} = \begin{bmatrix} 1 & (\mathbf{Ac})^\top \\ \mathbf{Ac} & \mathbf{AA}^\top \end{bmatrix}.$$

It follows that

$$\mathbf{P}^k \boldsymbol{\gamma} = \begin{bmatrix} 1 & (\mathbf{Ac})^\top \\ \mathbf{Ac} & \mathbf{AA}^\top \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \vdots \\ \gamma_{m+1} \end{bmatrix} = \gamma_1 \begin{bmatrix} 1 \\ \mathbf{Ac} \end{bmatrix} + \begin{bmatrix} (\mathbf{Ac})^\top \hat{\boldsymbol{\gamma}} \\ \mathbf{AA}^\top \hat{\boldsymbol{\gamma}} \end{bmatrix},$$

where $\hat{\gamma} = [\gamma_2 \cdots \gamma_{m+1}]^T$. Equation (10) now becomes

$$\gamma_1 \begin{bmatrix} 1 \\ \mathbf{Ac} \end{bmatrix} + \begin{bmatrix} (\mathbf{Ac})^T\hat{\gamma} \\ \mathbf{AA}^T\hat{\gamma} \end{bmatrix} = \begin{bmatrix} e_1 \\ \vdots \\ e_{m+1} \end{bmatrix}, \tag{12}$$

which implies that

$$\gamma_1 = e_1 - (\mathbf{Ac})^T\hat{\gamma} \tag{13}$$

and

$$\gamma_1\mathbf{Ac} + \mathbf{AA}^T\hat{\gamma} = \hat{\mathbf{e}}, \tag{14}$$

where $\hat{\mathbf{e}} \equiv [e_2 \cdots e_{m+1}]^T$. From (14) we have

$$\hat{\gamma} = (\mathbf{AA}^T)^{-1}\hat{\mathbf{e}} - \gamma_1(\mathbf{AA}^T)^{-1}\mathbf{Ac}. \tag{15}$$

Now substitute (15) into (13) to yield γ_1:

$$\gamma_1 = \frac{[e_1 - (\mathbf{Ac})^T(\mathbf{AA}^T)^{-1}\hat{\mathbf{e}}]}{[1 - (\mathbf{Ac})^T(\mathbf{AA}^T)^{-1}\mathbf{Ac}]}. \tag{16}$$

The direction finding map in (11) may be written as

$$\mathbf{s} = -(\gamma_1\mathbf{c} + \mathbf{A}^T\hat{\gamma}),$$

or using (15),

$$\mathbf{s} = -\gamma_1\mathbf{c} - \mathbf{A}^T(\mathbf{AA}^T)^{-1}[\hat{\mathbf{e}} - \gamma_1\mathbf{Ac}]. \tag{17}$$

If $\hat{\mathbf{e}} = \mathbf{0}$, or in other words $\mathbf{e} = [1\,0 \cdots 0]^T$, then expression (17) reduces to

$$\mathbf{s} = -\gamma_1(\mathbf{c} - \mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{Ac}),$$

which is simply a constant multiple of (6). The constant γ_1 must still be shown to be positive. To this end, note that if $\hat{\mathbf{e}} = \mathbf{0}$, then γ_1 is given by

$$\gamma_1 = \frac{e_1}{[1 - (\mathbf{Ac})^T(\mathbf{AA}^T)^{-1}\mathbf{Ac}]}. \tag{18}$$

The denominator in (18) can be written as $1 - \mathbf{c}^T[\mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{Ac}]$, where the second term is the inner product of $\mathbf{c}$ with the orthogonal projection of $\mathbf{c}$ onto the row space of $\mathbf{A}$, $\mathfrak{R}(\mathbf{A})$. Since $\mathbf{c}$ is normalized and $\mathbf{c} \notin \mathfrak{R}(\mathbf{A})$ by assumption, it follows that

$$\mathbf{c}^T[\mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{Ac}] \leq \|\mathbf{c}\| \cdot \|\mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{Ac}\| = \|\mathbf{A}^T(\mathbf{AA}^T)^{-1}\mathbf{Ac}\| < 1.$$

The denominator in (18) is therefore positive, which in turn shows that $\gamma_1 > 0$. The constant γ_1 therefore plays no role in practice if $\hat{\mathbf{e}} = \mathbf{0}$, and is not computed in this case as $\mathbf{s}$ is always normalized. This completes the proof. ∎

Theorem 1 shows that the search directions of Rosen's method are a special case of the *feasible descent cone directions* defined by (9) and (10)—the Rosen search directions are obtained by choosing the $\mathbf{e}$-vector $\mathbf{e} = [1\,0 \cdots 0]^T$ in (10). The more general search directions of the new method are generated by using other suitable choices for $\mathbf{e}$ in (10). Note that (16) and (17) are equivalent to formulae (10) and (9) and may alternatively be used in computation.

It is necessary to revisit the two assumptions of Theorem 1, for a new search direction cannot be calculated if the assumptions do not hold true:

(i) If $\mathbf{A}^k$ is not of full rank, then $(\mathbf{AA}^T)^{-1}$ is not defined in (15);

(ii) If $\mathbf{c} \in \mathfrak{R}(\mathbf{A}^k)$, then the denominator in (16) is zero and γ_1 is not defined.

It is easy to show that these scenarios correspond to a singular $\mathbf{P}^k$ matrix in (10). The two situations will now be discussed in turn in the next two sections, and strategies to deal with them will be presented. These strategies are further improvements over the algorithm of Snyman [6], where the algorithm is restarted from scratch each time the $\mathbf{P}^k$ matrix becomes singular.

7. STRATEGY TO DEAL WITH $\mathbf{c} \in \mathfrak{R}(\mathbf{A}^k)$

If $\mathbf{c} \in \mathfrak{R}(\mathbf{A}^k)$, it follows that $-\mathbf{c} = (\mathbf{A}^k)^T\boldsymbol{\lambda}$ for some $\boldsymbol{\lambda} \in \mathbb{R}^m$, called the multiplier vector. If all the components of $\boldsymbol{\lambda}$ corresponding to inequality constraints are nonnegative, then $W(\mathbf{x}^{k+1})$ contains a potentially optimal set of constraints. In other words, if a feasible point exists where all the constraints in $W(\mathbf{x}^k)$ are active, then that point is an optimal solution. If, on the other hand, some multipliers corresponding to inequality constraints are negative, then the constraint corresponding to the most negative multiplier may be discarded from the working set as in Rosen's method. To ensure that the next search direction does not violate or encounter the discarded constraint, a restriction must be placed on the choice of $\mathbf{e}$ when calculating the search direction in the next iteration. This restriction is given by the following theorem.

THEOREM 2. *Assume* $-\mathbf{c} = (\mathbf{A}^k)^T\boldsymbol{\lambda}$ *where* $\mathbf{A}^k$ *is a* $(m \times n)$ *matrix with* $m < n$. *Assume for convenience that* $\lambda_m < 0$, *associated with the inequality constraint* $(\mathbf{a}^m)^T\mathbf{x} \leq b_m$, *is the most negative of the components of the multiplier vector associated with the inequality constraints. If constraint* m *is discarded from the working set and a new search direction is calculated using (10) and (9) or (16) and (17), then constraint* m *will not be violated provided the* $\mathbf{e}$*-vector used in the calculation satisfies*

$$\sum_{i=n_e+1}^{m-1} e_{i+1}\lambda_i \geq -1.$$

PROOF. It is easy to show that the new search direction $\mathbf{s}$ is not zero. From (8), it follows that

$$-\mathbf{c}^T\mathbf{s} = e_1 = 1, \tag{19}$$

but

$$-\mathbf{c} = (\mathbf{A}^k)^T\boldsymbol{\lambda},$$

and therefore

$$-\mathbf{c}^T\mathbf{s} = \boldsymbol{\lambda}^T\mathbf{A}^k\mathbf{s}. \tag{20}$$

Combining (19) and (20) gives

$$\boldsymbol{\lambda}^T\mathbf{A}^k\mathbf{s} = e_1 = 1. \tag{21}$$

Let $\bar{\mathbf{A}}^k$ denote the matrix obtained by dropping $\mathbf{a}^m$ from $\mathbf{A}^k$. From (7) and (8),

$$-\bar{\mathbf{A}}^k\mathbf{s} = [0 \cdots 0\ e_{n_e+2} \cdots e_m]^T.$$

Equation (21) now becomes

$$-\sum_{n_e+1}^{m-1} e_{i+1}\lambda_i + \lambda_m(\mathbf{a}^m)^T\mathbf{s} = 1. \tag{22}$$

Since $\lambda_m < 0$ and the e_{i+1}'s are positive, equation (22) shows that $(\mathbf{a}^m)^T\mathbf{s} < 0$ if

$$\sum_{i=n_e+1}^{m-1} e_{i+1}\lambda_i \geq -1, \tag{23}$$

which completes the proof. ∎

Theorem 2 shows that the vector $\mathbf{e}$ in (10) can be chosen to ensure that the new search direction $\mathbf{s}$ does not lead to a violation of the discarded constraint. For example, one may choose $e_{i+1} = 0$ in (23) if $\lambda_i < 0$.

8. STRATEGY TO DEAL WITH RANK DEFICIENCY OF $\mathbf{A}^k$

Assume that the matrix $\mathbf{A}^k$ is not of full rank. In other words, the gradients of the constraints in $W(\mathbf{x}^k)$ are linearly dependent. This possibility is unique to the new method if $e_j > 0$ is used for $j = n_e + 2, \ldots, m + 1$, and cannot arise by using the search directions of Rosen's method, which have been shown to correspond to $\mathbf{e} = [1\,0 \cdots 0]^\top$. This observation, which is proved in the following theorem, suggests a strategy for dealing with the linear dependence of the gradients of the working set constraints, should it arise.

THEOREM 3. *Assume a current feasible point $\mathbf{x}^k$ with associated working set $W(\mathbf{x}^k)$ and that the matrix $\mathbf{A}^k = [\mathbf{a}^1 \cdots \mathbf{a}^m]^\top$ is of full rank. If the choice $\mathbf{e} = [1\,0 \cdots 0]^\top$ is used to calculate the next search direction, it follows from Theorem 1 that this search direction is given by*

$$\mathbf{s} = -[\mathbf{I} - (\mathbf{A}^k)^\top [\mathbf{A}^k (\mathbf{A}^k)^\top]^{-1} \mathbf{A}^k]\mathbf{c}.$$

If the nearest constraint is now encountered by taking a step in the direction of $\mathbf{s}$, and $\mathbf{A}^k$ is updated to $\mathbf{A}^{k+1}$, then $\mathbf{A}^{k+1}$ is also of full rank.

PROOF. Assume that the encountered constraint is given by $(\mathbf{a}^{m+1})^\top \mathbf{x} \leq b_{m+1}$. The matrix $\mathbf{A}^{k+1}$ is now given by $\mathbf{A}^{k+1} = [\mathbf{a}^1 \cdots \mathbf{a}^m\, \mathbf{a}^{m+1}]^\top$. Assume that $\mathbf{A}^{k+1}$ is not of full rank. Since $\mathbf{A}^k$ is of full rank, it follows that $\mathbf{a}^{m+1} \in \mathfrak{R}(\mathbf{A}^k)$. But $\mathbf{s} \in N(\mathbf{A}^k) \equiv \mathfrak{R}(\mathbf{A}^k)^\perp$; in other words, $\mathbf{s}^\top \mathbf{a}^{m+1} = 0$, which shows that $\mathbf{s}$ is "parallel" to the encountered constraint plane. This contradiction completes the proof. ∎

Theorem 3 suggests a strategy for the new method to deal with rank deficiency of $\mathbf{A}^k$ that can be formulated as follows:

(i) Drop the last encountered constraint from the working set and discard the corresponding row of $\mathbf{A}^k$. The resulting $\mathbf{A}$ matrix must be of full rank.

(ii) Calculate the new search direction by using $\mathbf{e} = [1\,0 \cdots 0]^\top$. This ensures that the discarded constraint cannot be encountered again. Note that the resulting path is not necessarily identical to the Rosen path, since all the constraints in $W(\mathbf{x}^k)$ need not be active.

9. CHOICE OF THE PARAMETER e

If no other preferential *a priori* choice of $\mathbf{e}$ is available, and a search direction through the interior of the feasible polytope is required, it is natural to choose the nonzero components of $\mathbf{e}$ as equal to one. This leads to the definition $\mathbf{e}^{1-0-1} \equiv [1\,0 \cdots 0\,1 \cdots 1]^\top$, where the zeros correspond to the equality constraints, and the 1's following the zeros correspond to the inequality constraints in the current working set. If the projected steepest descent direction (6) is required, the appropriate choice is $\mathbf{e} = [1\,0 \cdots 0]^\top$, which will be referred to as $\mathbf{e}^{1-0}$. If no equality constraints are present, then $\mathbf{e}^{1-0-1}$ can be replaced by $\mathbf{e}^1 \equiv [1 \cdots 1]^\top$.

10. THE FINAL ALGORITHM

We are now in the position to formalize the above ideas in a *feasible descent cone* algorithm.

ALGORITHM FDC. Assume a feasible starting point $\mathbf{x}^0$ given. Initialize $\mathbf{e}$ as $\mathbf{e} = \mathbf{e}^{1-0-1}$. If no equality constraints are present, replace $\mathbf{e}^{1-0-1}$ by $\mathbf{e}^1$ in the statement of the algorithm.

`Start of cycle`

(1) Set $i = 0$ and $W(\mathbf{x}^0) = I$. Construct $\mathbf{A}^0$ if I is not empty.

(2) Compute the new search direction $\mathbf{s}^i$. If $W(\mathbf{x}^0)$ is the empty set, then $\mathbf{s}^0 = -\mathbf{c}/\|\mathbf{c}\|$.

(3) Move in direction $\mathbf{s}^i$ until the nearest constraint (say $(\mathbf{a}^j)^\top \mathbf{x} \leq b_j$) is met at $\mathbf{x}^{i+1}$.

(4) Add constraint j to $W(\mathbf{x}^i)$ to give $W(\mathbf{x}^{i+1})$ and add the row $(\mathbf{a}^j)^\top$ to $\mathbf{A}^i$ to give $\mathbf{A}^{i+1}$. Set $i = i + 1$. If $\mathbf{A}^i$ has n rows, go to Step 8.

(5) If $\mathbf{e} = \mathbf{e}^{1-0-1}$, then determine if $\mathbf{A}^i$ is of full rank:
- If so, execute Step 6;
- If not, go to Step 7.

(6) {$\mathbf{A}^i$ `is of full rank`}
If $\mathbf{c} \in \mathfrak{R}(\mathbf{A}^i)$, then
Obtain the multiplier vector $\boldsymbol{\lambda}$, where $\boldsymbol{\lambda} = -[\mathbf{A}^i(\mathbf{A}^i)^\top]^{-1}\mathbf{A}^i\mathbf{c}$.
- If $\lambda_j \geq 0$ $\forall j \in W(\mathbf{x}^i) \cap J$, then $W(\mathbf{x}^i)$ contains a potentially optimal set of constraints:
  - If $\mathbf{e} = \mathbf{e}^{1-0}$, then the constraints in $W(\mathbf{x}^i)$ are active and $\mathbf{x}^i$ is optimal—`STOP`;
  - If $\mathbf{e} = \mathbf{e}^{1-0-1}$, set $\mathbf{e} = \mathbf{e}^{1-0}$, $\mathbf{x}^0 = \mathbf{x}^i$ and go to Step 1. {If the same working set is constructed in the next cycle, then an optimal solution is obtained.}
- If $\lambda_j < 0$ for some $j \in W(\mathbf{x}^i) \cap J$, then discard the constraint associated with the most negative multiplier from $W(\mathbf{x}^i)$ and $\mathbf{A}^i$. Choose an $\mathbf{e}$-vector that satisfies (23) and go to Step 2.

Else {$\mathbf{c} \notin \mathfrak{R}(\mathbf{A}^i)$}
Go to Step 2.
Endif

(7) {$\mathbf{A}^i$ `is not of full rank`} Drop the last encountered constraint from $W(\mathbf{x}^i)$ and discard the last row of $\mathbf{A}^i$. Set $\mathbf{e} = \mathbf{e}^{1-0}$ {i.e., adopt the Rosen search directions} and go to Step 2.

(8) {`The current working set contains` n `constraints`} Solve for the vertex $\mathbf{x}^v$ in $\mathbf{A}^i\mathbf{x}^v = \mathbf{b}^i$, where $\mathbf{b}^i$ represents the right-hand side vector for the constraints in $W(\mathbf{x}^i)$. If the solution process fails because $\text{rank}(\mathbf{A}^i) < n$, go to Step 7.
Obtain the multiplier vector $\boldsymbol{\lambda}$ from $\boldsymbol{\lambda} = -[(\mathbf{A}^i)^\top]^{-1}\mathbf{c}$
If $\mathbf{x}^v$ is feasible and $f(\mathbf{x}^v) \leq f(\mathbf{x}^i)$ then:
- If $\lambda_j \geq 0$ $\forall j \in W(\mathbf{x}^i) \cap J$, then $\mathbf{x}^v$ is optimal—`STOP`.
- If $\lambda_j < 0$ for some $j \in W(\mathbf{x}^i) \cap J$, then discard the constraint associated with the most negative multiplier from $W(\mathbf{x}^i)$ and $\mathbf{A}^i$. Choose an $\mathbf{e}$-vector that satisfies (23), set $\mathbf{x}^i = \mathbf{x}^v$, and go to Step 2.

Else {$\mathbf{x}^v$ `infeasible or` $f(\mathbf{x}^v) > f(\mathbf{x}^i)$}
- If $\lambda_j \geq 0$ $\forall j \in W(\mathbf{x}^i) \cap J$ then $W(\mathbf{x}^i)$ contains a potentially optimal set of constraints. Set $\mathbf{e} = \mathbf{e}^{1-0}$, $\mathbf{x}^0 = \mathbf{x}^i$ and go to Step 1.
- If $\lambda_j < 0$ for some $j \in W(\mathbf{x}^i) \cap J$, then discard the constraint associated with the most negative multiplier from $W(\mathbf{x}^i)$ and $\mathbf{A}^i$. Choose an $\mathbf{e}$-vector that satisfies (23) and go to Step 2.

Endif

`End of cycle`

11. CALCULATION OF THE SEARCH DIRECTIONS

Formulae (16) and (17) are used in the numerical calculation of the feasible search directions. This may be done economically by updating the matrix $(\mathbf{A}\mathbf{A}^\top)^{-1}$ in each time the working set is changed. This updating procedure is normally used in Rosen's method and is described in [13]. The procedure shows whether or not the new matrix $\mathbf{A}$ remains of full rank if a constraint is added to the working set (Step 5 of Algorithm FDC). Furthermore, it is easy to show that the denominator in (16) becomes zero if $\mathbf{c} \in \mathfrak{R}(\mathbf{A}^k)$ (Step 6 of Algorithm FDC). More details may be found in [14].

12. RESULTS OF COMPUTER IMPLEMENTATION

The FDC algorithm was evaluated for three classes of LP problems, which will be discussed in turn:

(1) Random LP problems;

(2) Weight minimization of plane trusses, formulated as LP problems; and

(3) LP test problems from the `Netlib` set [15].

12.1. Results for Random LP Problems

The problem under consideration is

$$\min\left[-\sum_{i=1}^{n} x_i\right],$$

subject to

$$\sum_{j=1}^{n} a_{ij}x_j \leq 10, \quad i = 1,\ldots,m$$

and

$$x_j \geq 0, \quad j = 1,\ldots,n,$$

with each a_{ij} randomly selected between 0 and 1. The most important difference between random and real-life LP problems is that the constraint matrices for real-life problems are usually very sparse, as opposed to the dense matrices of random problems. Furthermore, unlike real-life problems, random problems are generally not ill-conditioned or degenerate. Table 1 gives the performance of the FDC algorithm for random problems of various sizes as compared to the performance of the `IMSL` simplex routine `DDLPRS` [16]. The entries correspond to the ratio

$$r = \frac{\text{Average CPU time required by FDC}}{\text{Average CPU time required by DDLPRS}},$$

where the average is taken over a set of twenty random problems. The entries in brackets give the average number of iterations performed by the FDC algorithm.

Table 1. Performance of algorithm FDC on random problems.

	n 20	50	100	150
m 200	0.31 (58.9)	0.39 (161)	0.69 (369)	1.10 (533)
150	0.37 (56.2)	0.52 (158)	1.02 (323)	1.96 (536)
100	0.60 (53.3)	0.90 (142)	1.68 (291)	
50	0.79 (48.6)	1.26 (128)		
20	0.91 (45.5)			

It is clear from Table 1 that the FDC algorithm is superior to the simplex algorithm for entries sufficiently removed from the diagonal $n = m$. In fact, for $m \gg n$ the performance of the FDC algorithm is significantly better. On the other hand, the FDC algorithm is slower than the simplex method along the diagonal, and this effect is amplified as $m = n$ increases. These observations are consistent with the findings of Snyman [6], but a comparison shows that the new algorithm FDC is significantly faster than its predecessor. Also note that the iterations required for convergence by FDC increases very slowly for fixed n with increasing m. As discussed in [6], this is an indication of suitability for parallel implementation.

Finally, it is important to note that the algorithm FDC proved to be a robust algorithm—hundreds of random problems were solved to compile Table 1 and the FDC algorithm never failed to terminate at the optimal solution. Unfortunately, the random problems are of limited importance, but the performance of FDC on this class of problems is nevertheless encouraging.

12.2. Results for Truss Problems

The test problems in this section involve the weight minimization of plane trusses, formulated as LP problems. A description of the problems as well as a problem generator is available from `Netlib`. Four different trusses of increasing size were examined. These trusses have two fixed base nodes supporting stacked rows of eleven nodes each. Further details may be found in [14].

Table 2. Performance of FDC algorithm on truss problems.

Nodes	n	m	FDC	DDLPRS	Ratio	FDC iter.
22	44	114	0.68 s	0.63 s	1.1	107
33	66	196	3.88 s	1.96 s	1.98	418
44	88	278	8.67 s	3.93 s	2.21	522
55	110	360	19.25 s	7.74 s	2.49	769

Table 2 gives the results for the trusses with 2–5 rows of free nodes. The first column gives the number of free nodes for the problem under consideration. The columns marked n and m give the number of variables and constraints of the dual minimization problem respectively. The next two columns give the CPU times required by the FDC algorithm and the simplex algorithm DDLPRS respectively, and the column marked 'ratio' gives the ratio of the two CPU times. The ratio is defined as in the previous section. The final column gives the number of iterations performed by the FDC algorithm to obtain convergence—the FDC algorithm terminated at an optimal solution for each of the problems in Table 2.

It should be noted that the primal problem was solved by the simplex method DDLPRS to obtain the entries in Table 2, whereas the dual problem was solved by FDC. The dual problem was solved, since the results for the random problems showed that the FDC algorithm generally performs better if $m > n$, which is the case here for the dual problem.

The FDC algorithm is slower than the simplex algorithm for the problems in Table 2, but is not overshadowed, although the relative performance of the FDC algorithm deteriorates slightly with increasing problem size. Once again, the FDC algorithm terminated at an optimal solution for each problem. The overall performance of FDC on the truss problems is therefore robust and fairly competitive. Moreover, the existing computer implementation of the algorithm can no doubt be made more efficient.

12.3. Results for the Netlib Test Problems

Finally, six of the smaller problems from the Netlib test set were used as test problems for the FDC algorithm. Table 3 gives information on the problems as well as the performances of both the FDC and DDLPRS codes.

Table 1. Performance of the FDC algorithm on `Netlib` problems.

Name	Rows	Columns	Nonzero	No. eq.	DDLPRS	FDC	FDC iter.
`Afiro`	28	32	88	9	0.30 s	0.39 s	55
`Adlittle`	57	97	465	15	0.96 s	12.0 s	789
`Sc105`	106	103	281	45	1.71 s	2.42 s	149
`Share2b`	97	79	730	13	1.82 s	46.2 s	2750
`Blend`	75	83	521	43	1.31 s	fails	—
`Stocfor1`	118	111	474	62	1.61 s	fails	—

The column marked 'Nonzero' gives the number of nonzero entries in the constraint matrix, and the column 'No. eq.' gives the number of equality constraints for the respective problems. The column 'FDC iter.' gives the number of iterations performed by the FDC method to obtain convergence.

The first observation from Table 3 is that the FDC algorithm only solved four of the six problems successfully. The FDC algorithm terminated at the optimal solution for the first three problems in the table, but not for the problem `Share2b`, where slow asymptotic convergence took place. The algorithm failed for the problem `Blend`, where no progress was made beyond the degenerate point $[0 \cdots 0]^\top$. The algorithm also failed for the problem `Stocfor1`, where repeated ill-conditioning of the matrix $(\mathbf{A}\mathbf{A}^\top)^{-1}$ prevented normal execution.

In defense of the FDC algorithm, one must note that it is competitive with the simplex method for the problems where no ill-conditioning was encountered, namely `Afiro` and `Sc105`. Some ill-conditioning was found for the problem `Adlittle`, but this did not prevent termination at the optimal solution. However, more sophisticated numerical implementation may well overcome the current difficulties with ill-conditioning.

13. CONCLUSION

The original interior feasible direction method of Snyman has been improved to allow for search directions within a feasible descent cone at each iteration. Theoretical analysis of the modified method shows that it is a generalization of Rosen's classic gradient projection method. The more general method allows for descent steps to be taken through the interior as well as along the boundary of the feasible polytope. This is in contrast to Rosen's method where, except for the first iteration, all steps are restricted to the boundary. The FDC method, therefore, has the potential for taking large steps through the interior with resultant fast convergence.

The FDC method outperforms Snyman's original method when applied to random problems. For the latter problems, the numerical results also indicate that the new method is consistently superior to the simplex method in the case where the number of constraints is much greater than the number of variables. Moreover, the new method is robust and does well on real-life LP problems involving weight minimization of plane trusses. Considering that one may reasonably expect future improvements in the efficiency of implementation of the FDC method, the performance of the current version on truss problems is surprisingly competitive. In contrast, the performance on the challenging small `Netlib` problems is relatively disappointing, where the method fails to solve two of the more ill-conditioned problems.

As mentioned in the introduction, the potential of the FDC approach for solving nonlinear problems has already been demonstrated by the successful application of a simplified and relatively crude FDC algorithm to nonlinear truss problems in structural optimization [7].

With this in mind, the greatest value of this study has been the degree of refinement of the method achieved with reference to its application to the more simple LP problem. In particular, great attention has been paid to the efficient computation of search directions, and the economic alternative strategies to deal with complications such as linear dependence of constraints. Having done this successfully, specifically with respect to the LP truss problems, these refinements may confidently be applied to the more general and more important nonlinear truss problems. Here, we may expect significant improvement in efficiency compared to that previously achieved, with the FDC method hopefully coming into its own.

REFERENCES

1. J.B. Rosen, The gradient projection method for nonlinear programming, Part I, Linear constraints, *SIAM J. Appl. Math.* **8**, 181–217 (1960).
2. J.B. Rosen, The gradient projection method for nonlinear programming, Part II, Nonlinear constraints, *SIAM J. Appl. Math.* **8**, 514–553 (1961).

3. D. Goldfarb, Extension of Davidson's variable metric method to maximization under linear equality and inequality constraints, *SIAM J. Appl. Math.* **17**, 739–764 (1969).
4. D.-Z. Du and X.-S. Zhang, A convergence theorem of Rosen's gradient projection method, *Math. Prog.* **36**, 135–144 (1986).
5. D.-Z. Du and X.-S. Zhang, Global convergence of Rosen's gradient projection method, *Math. Prog.* **44**, 356–366 (1989).
6. J.A. Snyman, An interior feasible direction method with constraint projections for linear programming, *Computers Math. Applic.* **20** (12), 43–54 (1990).
7. J.A. Snyman and N. Stander, A new first-order interior feasible direction method for structural optimization, *Int. J. Numer. Methods Eng.* **36**, 4009–4025 (1993).
8. G.B. Dantzig, *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ (1963).
9. W.I. Zangwill, The convex simplex method, *Mgmt. Sci.* **14**, 221–283 (1967).
10. G. Mitra, M. Tamiz and J. Yadegar, Experimental investigation of an interior search method within a simplex framework, TR/06/86, Brunel University, Uxbridge, UK (1986).
11. N. Karmarkar, A new polynomial time algorithm in linear programming, *Combinatorica* **4**, 373–395 (1984).
12. C. Roos and J.-Ph. Vial, A polynomial method of approximate centers for linear programming, *Math. Prog.* **54**, 295–305 (1992).
13. M.S. Bazaraa, H.D. Sherali and C.M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 2nd Edition, John Wiley and Sons (1993).
14. E. de Klerk and J.A. Snyman, FDC: An improved algorithm for linearly constrained problems in structural optimization, Report 93-4, Structural Optimization Research Group, University of Pretoria, Pretoria, South Africa (1993).
15. J.J. Dongarra and E. Grosse, Distribution of mathematical software via electronic mail, *SIGNUM Newsletter* **20**, 45–47 (1985).
16. *International Mathematical and Statistical Libraries*, 9th Edition, IMSL, Houston, TX (1982).